Angular Universal

•••

Applying the studies to the Rhizom Traceability project

rhizom
IN AUTONOMY WE TRUST

The basics of the desirable web

- Findable
- Fast load
- Quick interaction

rhizom
IN AUTONOMY WE TRUST

Ok, but what's the problem with Angular?

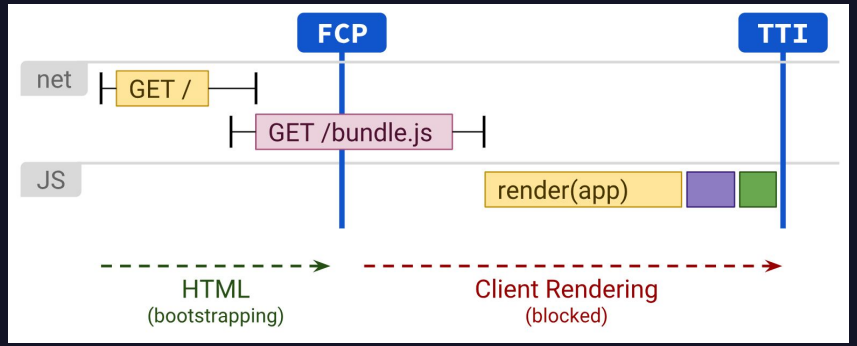
- Findable?
  - Crawlers
- Slow loading
- Slow interaction

Why use Angular Universal?

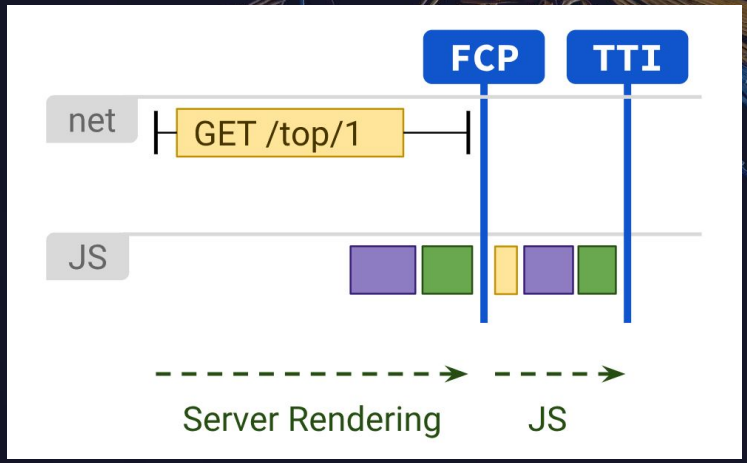
- SEO friendly
- Decrease first page load time
- Time to Interactive (TTI)

Angular CLI vs Angular Universal

- CSR: Client-Side Rendering



- SSR: Server-Side Rendering



rhizom
IN AUTONOMY WE TRUST

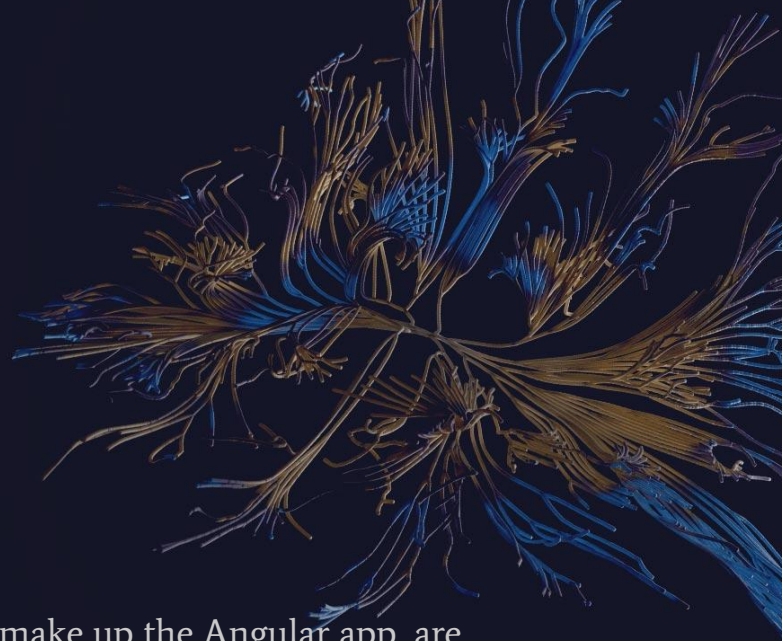
Differences

- Server is responsible for the initial rendering
- Using NodeJS with ExpressJS
- There is no DOM api

Challenges

- Lack of DOM resources (server state)
    - window, document, localStorage, indexedDB, setTimeout* and setInterval*
- Maintaining a NodeJS server with ExpressJS
- Third party libraries
- Adapt the team to the new

Default lifecycle



1. The user navigates to the application
2. The server serves up index.html
3. index.html is rendered in the DOM
4. The JavaScript and CSS files referenced in index.html, which make up the Angular app, are downloaded
5. The Angular application bootstraps in the browser and renders inside of app-root

CSR

```
<html lang="en">
    <head>
        ...
        <link rel="stylesheet" href="styles.3ff695c00d717f2d2a11.css">
    </head>
    <body>
        <app-root></app-root>
        <script type="text/javascript" src="runtime.ec2944dd8b20ec099bf3.js"></script>
        <script type="text/javascript" src="polyfills.3bfd66addbd0d2814591.js"></script>
        <script type="text/javascript" src="main.8d3bca5df2cd7b3fd5cf.js"></script>
    </body>
</html>
```

SSR lifecycle

1. The user navigates to the application
2. The server (a Node.js server) executes the application in order to render the page that the user navigated to, and then serves up index.html
3. index.html is rendered in the DOM
4. The JavaScript and CSS files referenced in index.html are downloaded
5. The Angular application bootstraps in the browser and renders inside of app-root, and takes over from the server rendered version of the app

SSR

```
<html lang="en">
    <head>
        ...
        <link rel="stylesheet" href="styles.3ff695c00d717f2d2a11.css">
        <style ng-transition="serverApp">
            [_nghost-serverApp-c0]{display:block}
            ...
        </style>
    </head>
    <body>
        <app-root>
            <div><h1>My application</h1></div>
            ...
        </app-root>
        <script type="text/javascript" src="runtime.ec2944dd8b20ec099bf3.js"></script>
        <script type="text/javascript" src="polyfills.3bfd66addbd0d2814591.js"></script>
        <script type="text/javascript" src="main.8d3bca5df2cd7b3fd5cf.js"></script>
    </body>
</html>
```

Angular Universal is a trick

- Only that
- Interactivity (CLI)
- but...

...but, it is necessary to study more

- Server and browser states
- Transfer states
- Lifecycle
- Improving performance (Change detector)
- Pre-rendering

Is implementation necessary?

- Yes, but it depends
- Not for applications with private routes
    - Changes in HTML, JS and CSS resolve
- Rhizom Framework is a case where it is necessary to apply (Urgently)
    - SEO, mobile

rhizom
IN AUTONOMY WE TRUST

That is all

Gesiel Rosa

linkedin.com/in/gesielrosa
github.com/gesielrosa

- https://willtaylor.blog/angular-universal-gotchas/
- https://willtaylor.blog/angular-universal-for-angular-developers/
- https://developers.google.com/web/updates/2019/02/rendering-on-the-web